NAME (from your UF ID): ________________________ **UF ID#:** ___________
(Please **PRINT**)

-------------------- CEN 4072/6070 Software Testing & Verification ------------------

Quiz 1 -- Spring 2017

You have 30 minutes to work on this exam. It is a "closed-book/closed-notes" test. Pay attention to point values, since you may not have time to work all 7 problems. PRINT your name and UF ID# above NOW and sign the pledge at the bottom of the last page, if appropriate, when you are finished. PLEASE PRINT **– do NOT write** *CURSIVELY* **–** ANSWERS IN THE SPACE PROVIDED ONLY, PREFERABLY USING A BALLPOINT PEN TO INCREASE LEGIBILITY. Good luck!

1. (3 pts.) Recall the Cause-Effect (C-E) Analysis test case selection (coverage criterion) strategy know as "Strategy #3":

```
REPEAT
    Select the next (initially, the first) Effect.
    Tracing back through the graph (right to left),
    find all feasible combinations of connected
    Cause values that result in the Effect being
    True.
    For each new such combination found:
        Determine values of all other Effects, and
        Enter values for each Cause and Effect in a
        new column of the test case coverage matrix.
UNTIL each Effect has been selected.
```

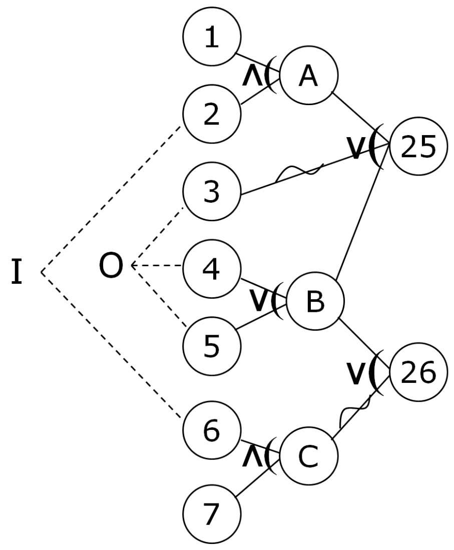
   Which one of the following correctly characterizes the general relationship between Strategy #3 and AFCCV ("All Feasible Combinations of Cause Values")? (Circle one only.)

   i. Strategy #3 and AFCCV are equivalent (each subsumes the other)
   [✓ ii. Strategy #3 subsumes AFCCV]
   iii. AFCCV subsumes Strategy #3
   iv. Strategy #3 and AFCCV are independent (neither criterion subsumes the other)
   v. (none of the above)

2. (5 pts.) Recall from Lecture Notes 6.1 ("Case Study: Black-Box Testing) the application of black-box test case design techniques to the power function **pow(x,y)**. Using a number line representing the value of x^y, clearly identify ALL region(s) and boundary values on the line associated with UNDERFLOW and OVERFLOW.

3. (3 pts.) The draft MAN page for **pow()** used in the black-box testing case study primarily described exceptional/error behaviors. What source and approach was used to identify different **nominal/non-error** behaviors that could be modeled?

4. Consider the following Cause-Effect graph.



a. (5 pts.) How many test cases would be required to achieve **AFCCV** (All Feasible Combinations of Cause Values) coverage? **AE** (All Effects) coverage? (Hint: I="at least one"; O="one and only one.")

Number of cases required: for AFCCV: _____ for AE: _____

b. (7 pts.) Identify all feasible combinations of connected Cause values that result in Effect 25 being true, ***except*** those eliminated by the culling rules discussed in class. Enter a test case template reflecting each such combination in the following test case coverage matrix. (Use only the number of columns needed.) Represent "don't care" Cause values as "*".

	TEST CASES									
CAUSES	1	2	3	4	5	6	7	8	9	10
(1)										
(2)										
(3)										
(4)										
(5)										
(6)										
(7)										
EFFECT										
(25)										

5. a. (5 pts.) Based on the MAN page given in Problem Set 1 for function **fmod(x,y)**, give the actual expected **fmod** return VALUES for each of the following examples:

fmod(3.5, 2.3) =	fmod(3.5, -2.3) =
fmod(-3.5, 2.3) =	fmod(-3.5, -2.3) =
fmod(3.5, 0) =	

b. (6 pts.) Five Effects were listed in the Solution Notes for **fmod(x,y)**. State the Effects which describe the general behavior of **fmod** and **matherr** that would apply to each of the following examples: (STATE THE EFFECTS AS GIVEN IN THE SOLUTION NOTES – do not just give the assigned number of the Effects or the specific values returned for the examples as in part (a) above.)

fmod(-3.5, 2.3):

fmod(-3.5, -2.3):

c. (2 pts.) The **sort** command from Problem Set 1 actually performs 3 distinct high-level functions. One is sorting all the lines of text contained in one or more input files. What are the other two?

d. (8 pts.) According to the given MAN page for **sort**, which of the following are valid options to override the default ordering rules of the sort command, and which are not? Indicate "valid" or "not valid" as appropriate. To compensate for random guessing, your score in points will be 2 times the number of [correct minus incorrect] answers, or 0 – whichever is greater. Therefore, if you are not more than 50% sure of your answer, consider skipping the item.

i. Compares as days of the week (SUN < MON < … < SAT)	valid	[✓ not valid]
ii. Compares characters "START_CHAR" (>=1) through "END_CHAR" (<=MAX_LINE_LENGTH) only	valid	[✓ not valid]
iii. Reverses the sense of comparisons	[✓ valid]	not valid
iv. Ignores trailing blank characters in a line	valid	[✓ not valid]

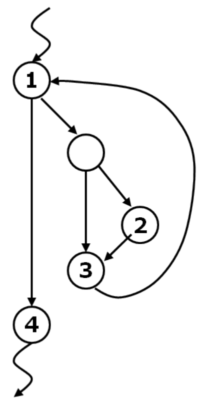
e. (3 pts.) Inputs for three test cases that *should* have been (but were NOT) included in the Solution Notes for command **sort** are:

```
sort -df infile
sort -r -o outfile infile1 infile2
sort -um infile  (where the lines of infile are already sorted)
```

Briefly state the simple rationale for this. (Hint: the rationale was mentioned at the end of Lecture Notes 6.1, Case Study: Black-Box Testing, in the context of testing function **pow()** using Intuition and Experience.)

6. Consider the following program and its control flow graph:

```
1. while p do
       if k then
2.         s1
3.     end_if_then
       s2
4. end_while_do
```



a. (2 pts.) Consider a test case with execution path <1,3,1,4>. Circle all of the following coverage criteria that would be met by executing ONLY this test case. (Circle "none" if none would be met.)

[✓ none]   statement   branch   condition   basis paths   path

b. (2 pts.) Consider a test case with execution path <1,2,3,1,4>. Circle all of the following coverage criteria that would be met by executing BOTH of these test cases. (Circle "none" if none would be met.)

none   statement   branch   condition   basis paths   path

c. (2 pts.) Consider a test case with execution path <1,4>. Circle all of the following coverage criteria that would be met by executing ALL THREE of these test cases. (Circle "none" if none would be met.)

none   statement   branch   condition   basis paths   path

d. (2 pt.) What is the *minimum* number of test cases that would be required, in general, to achieve **statement, branch, and condition coverage** for this program?

Circle one: none   1   2   3   4   infinite

e. (2 pt.) What is the *minimum* number of test cases that would be required, in general, to achieve **basis paths coverage** for this program?

Circle one: none   1   2   3   4   infinite

f. (2 pt.) What is the *minimum* number of test cases that would be required, in general, to achieve **path coverage** for this program?

Circle one: none   1   2   3   4   infinite

7. (8 pts.) Consider pseudocode program:

```
(1) input (A,B);
(2) if (A OR B) then
(3)      s1
(4) end_if
```

for Boolean variables A, B. PROVE, **using one or more counter-examples, as necessary**, that

path coverage and condition coverage are independent

Explicitly identify any counter-example(s) used in your proof by identifying the Boolean input values assumed for A and B, the path(s) sensitized, etc. Your counter-example(s) and explanation must make your conclusion clear.

On my honor, I have neither given nor received unauthorized aid on this exam and I pledge not to divulge information regarding its contents to those who have not yet taken it.

____
signature